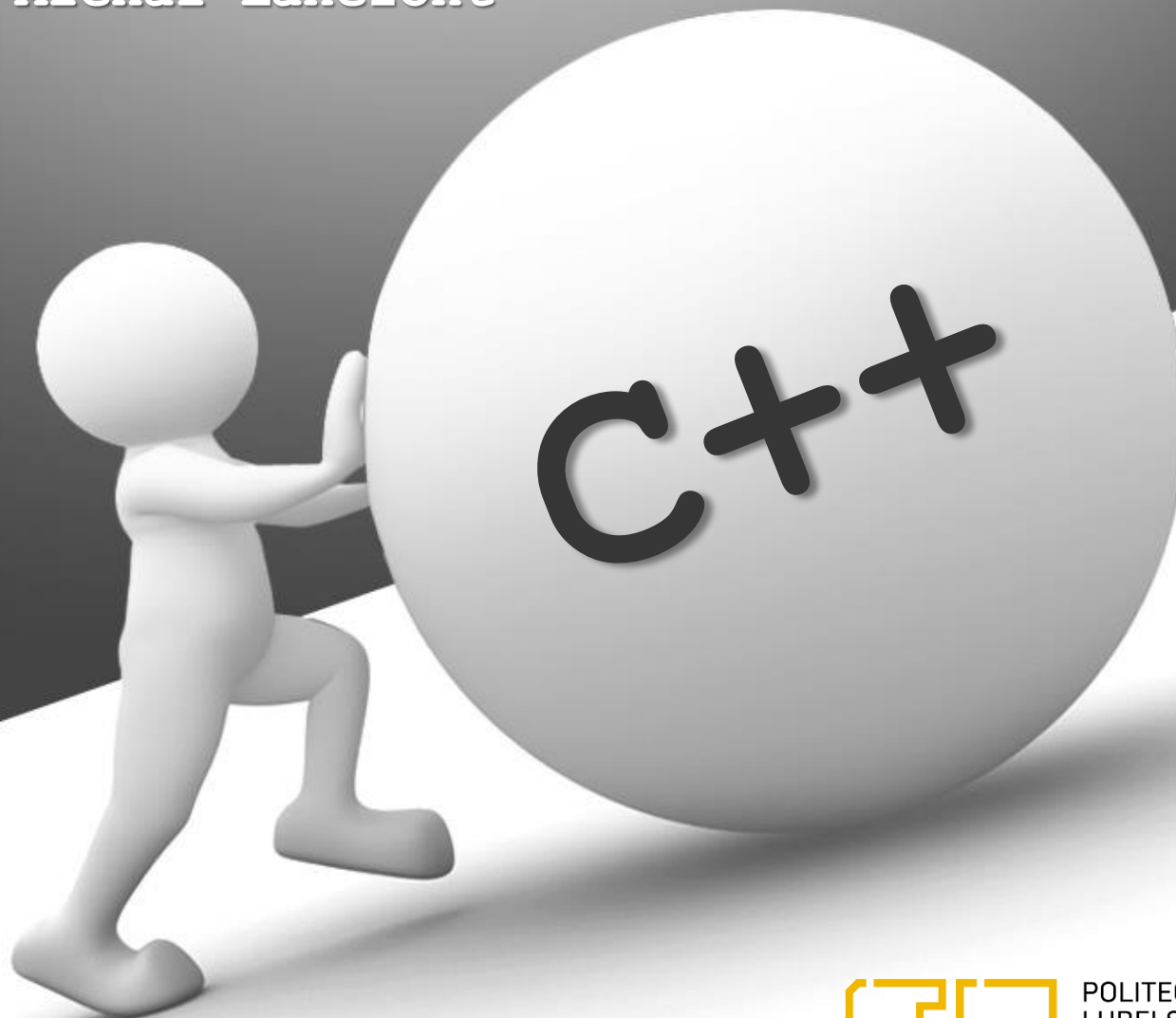


Informatyka II

dr inż. Michał Łanczont

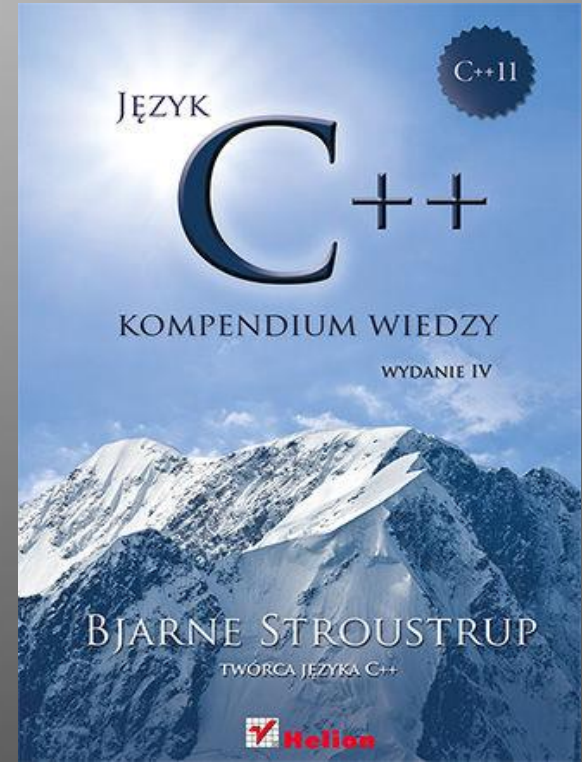


POLITECHNIKA
LUBELSKA
WYDZIAŁ ELEKTROTECHNIKI
I INFORMATYKI

Zaliczenie przedmiotu

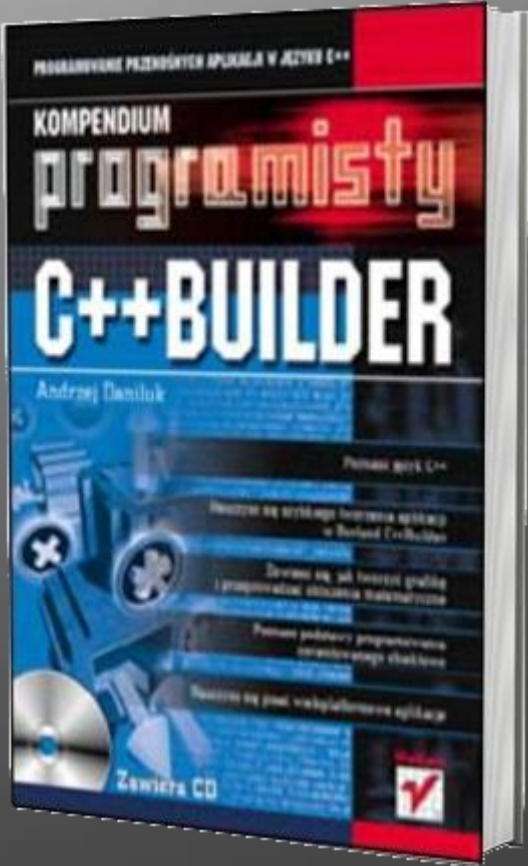
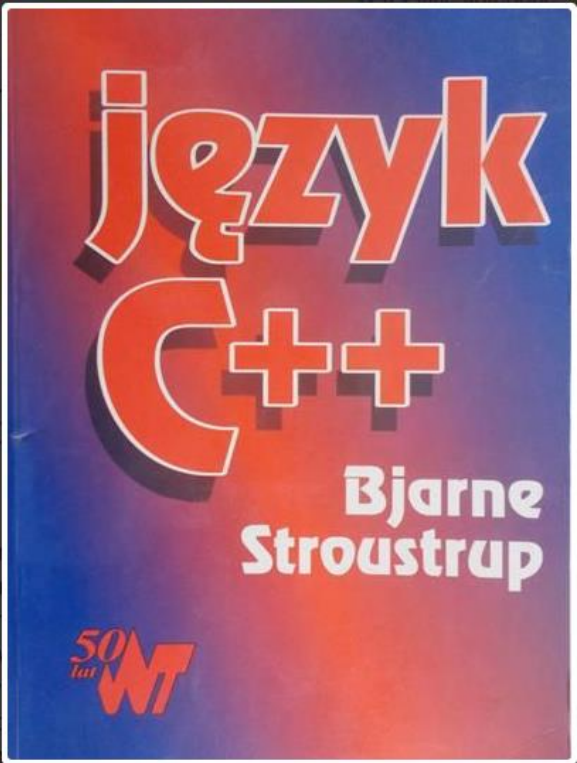
- Wykład – egzamin
 - Egzamin pisemny
 - Pierwszy termin w sesji letniej
 - W sesji egzamin poprawkowy I i II (wrześniowy)
- Laboratoria – Określone przez prowadzącego
 - Zaliczenie ćwiczeń cząstkowych
 - Zaliczenie końcowe na ostatnich zajęciach





- Stephen Prata, *Język C++. Szkoła programowania*. Wydanie VI, Helion
- Bjarne Stroustrup, *Język C++. Kompendium wiedzy*. Wydanie IV, Helion







Podstawy C++

TABLICE INFORMACYCZNE • Radostaw Bokół

WPROWADZENIE

Podstawy C++ to podręcznik dla początkujących i średniozaawansowanych programistów. Zawiera wszystkie niezbędne informacje o języku C++, jego składni, typach danych, strukturach danych, funkcjach, bibliotekach i narzędziach. Podręcznik jest przeznaczony dla osób, które chcą nauczyć się programowania w C++ i zastosować te umiejętności w praktyce.

Typy danych

Typ	Opis
int	całkowity
short	krótki całkowity
long	długi całkowity
float	liczba zmiennoprzecinkowa
double	liczba zmiennoprzecinkowa o podwójnej precyzji
char	znak
bool	wartość logiczna
void	brak typu

Struktury danych

Struktura	Opis
array	tablica
vector	wektor
list	lista
map	mapa
set	zbiór
string	ciąg znaków

Struktury danych

Struktura	Opis
array	tablica
vector	wektor
list	lista
map	mapa
set	zbiór
string	ciąg znaków

Struktury danych

Struktura	Opis
array	tablica
vector	wektor
list	lista
map	mapa
set	zbiór
string	ciąg znaków



C++

TABLICE INFORMACYCZNE

WPROWADZENIE

Podstawy C++ to podręcznik dla początkujących i średniozaawansowanych programistów. Zawiera wszystkie niezbędne informacje o języku C++, jego składni, typach danych, strukturach danych, funkcjach, bibliotekach i narzędziach. Podręcznik jest przeznaczony dla osób, które chcą nauczyć się programowania w C++ i zastosować te umiejętności w praktyce.

Typy danych

Typ	Opis
int	całkowity
short	krótki całkowity
long	długi całkowity
float	liczba zmiennoprzecinkowa
double	liczba zmiennoprzecinkowa o podwójnej precyzji
char	znak
bool	wartość logiczna
void	brak typu

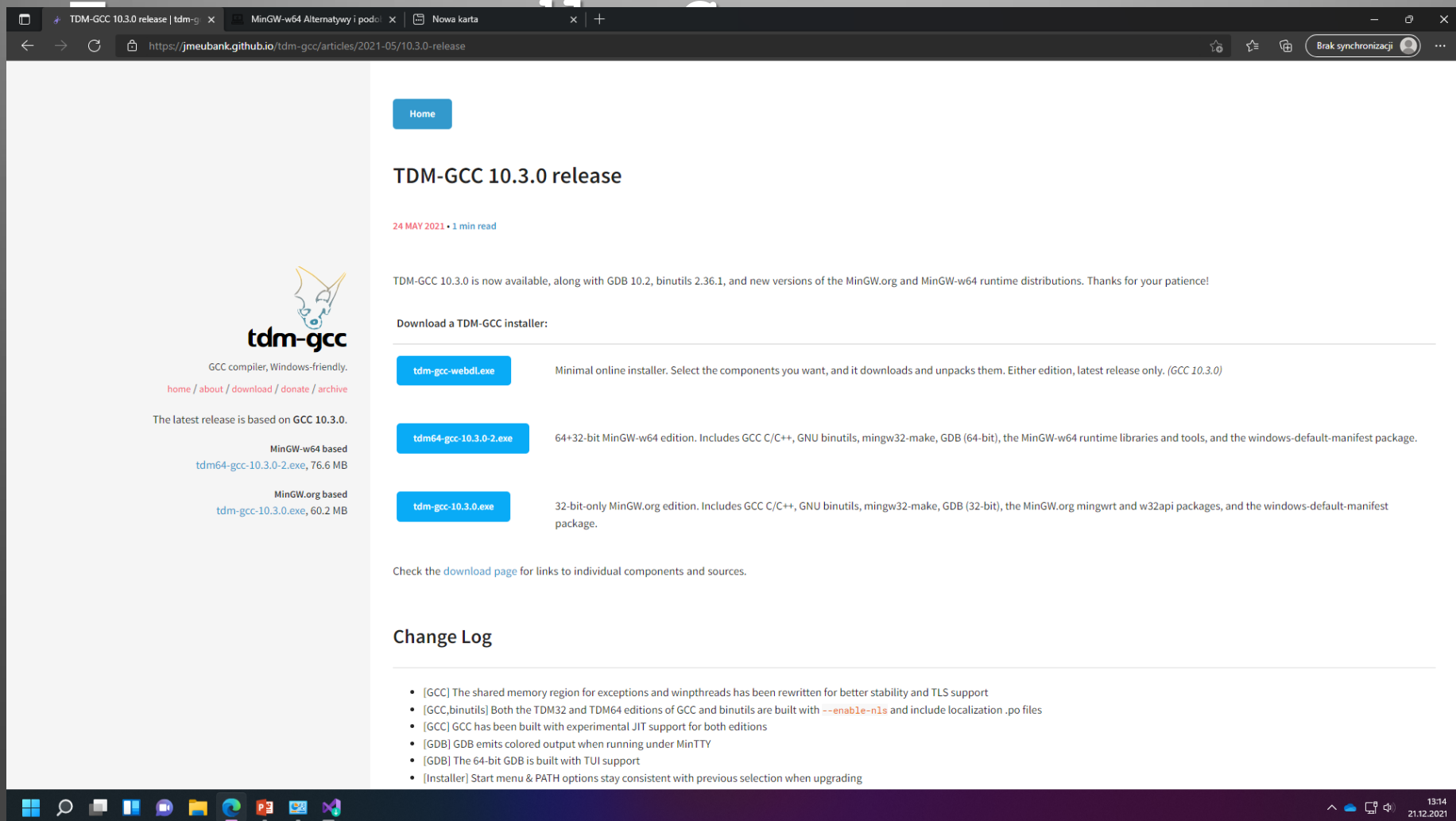
Struktury danych

Struktura	Opis
array	tablica
vector	wektor
list	lista
map	mapa
set	zbiór
string	ciąg znaków

Struktury danych

Struktura	Opis
array	tablica
vector	wektor
list	lista
map	mapa
set	zbiór
string	ciąg znaków





TDM-GCC 10.3.0 release

24 MAY 2021 • 1 min read

TDM-GCC 10.3.0 is now available, along with GDB 10.2, binutils 2.36.1, and new versions of the MinGW.org and MinGW-w64 runtime distributions. Thanks for your patience!

Download a TDM-GCC installer:

tdm-gcc-webdl.exe Minimal online installer. Select the components you want, and it downloads and unpacks them. Either edition, latest release only. (GCC 10.3.0)

tdm64-gcc-10.3.0-2.exe 64+32-bit MinGW-w64 edition. Includes GCC C/C++, GNU binutils, mingw32-make, GDB (64-bit), the MinGW-w64 runtime libraries and tools, and the windows-default-manifest package.

tdm-gcc-10.3.0.exe 32-bit-only MinGW.org edition. Includes GCC C/C++, GNU binutils, mingw32-make, GDB (32-bit), the MinGW.org mingwrt and w32api packages, and the windows-default-manifest package.

Check the [download page](#) for links to individual components and sources.

Change Log

- [GCC] The shared memory region for exceptions and winthreads has been rewritten for better stability and TLS support
- [GCC,binutils] Both the TDM32 and TDM64 editions of GCC and binutils are built with `--enable-nls` and include localization .po files
- [GCC] GCC has been built with experimental JIT support for both editions
- [GDB] GDB emits colored output when running under MinTTY
- [GDB] The 64-bit GDB is built with TUI support
- [Installer] Start menu & PATH options stay consistent with previous selection when upgrading

— MICROSOFT VISUAL STUDIO 2022

(Community – darmowa wersja)



Kompilator C/C++

Zmienne środowiskowe

Zmienne użytkownika dla mlanc

Zmienna	Wartość
MOZ_PLUGIN_PATH	C:\Program Files (x86)\Foxit Software\Foxit Reader\plugins\
OneDrive	C:\Users\mlanc\OneDrive
OneDriveConsumer	C:\Users\mlanc\OneDrive
Path	C:\MinGW\bin;C:\Users\mlanc\AppData\Local\atom\bin;
TEMP	C:\Users\mlanc\AppData\Local\Temp
TMP	C:\Users\mlanc\AppData\Local\Temp

Nowa...

Edytuj...

Usuń

Zmienne systemowe

Zmienna	Wartość
OMP_NUM_THREADS	4
OS	Windows_NT
P_SCHEMA	C:\Program Files (x86)\Solid Edge V20\Schema
Path	C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wb...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 94 Stepping 3. GenuineIntel

Nowa...

Edytuj...

Usuń

OK

Anuluj

Edycja zmiennej środowiskowej

C:\WINDOWS\system32

C:\WINDOWS

C:\WINDOWS\System32\Wbem

C:\WINDOWS\System32\WindowsPowerShell\v1.0\

C:\WINDOWS\System32\OpenSSH\

C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common

C:\Program Files\Microsoft SQL Server\120\Tools\Binn\

C:\WINDOWS\system32

C:\WINDOWS

C:\WINDOWS\System32\Wbem

C:\WINDOWS\System32\WindowsPowerShell\v1.0\

C:\WINDOWS\System32\OpenSSH\

C:\Program Files\PuTTY\

C:\Program Files\NVIDIA Corporation\NVIDIA NvDLISR

C:\MinGW\bin

Nowy

Edytuj

Przeglądaj...

Usuń

Przenieś w górę

Przenieś w dół

Edytuj tekst...

OK

Anuluj



Embarcadero Dev-C++ 6.3

PlikEdycjaSzukajWidokProjektUruchomNarzędziaAStyleOknoPomoc

TDM-GCC 9.2.0 64-bit Release

(globals)

ProjektKlasyOdpluskwiacz



WELCOME

Version 6.3
[View Documentation](#)

Hotkeys

Open	Ctrl + O
Save	Ctrl + S
Zoom	Ctrl + Scroll
Run	F10
Compile	F9
Clear	Ctrl + W

Nowy dokument

Otwórz dokument

Options

Zmien motyw

Zmien czcionkę

Zmien język

Lotto.dev
C:\Users\mlanc\Mój dysk\DYDAKTYKA\Informatyka\ILaboratorium\dev\Lotto.dev

Kompilator (2)ZasobyLog kompilacjiOdpluskwiaczWyniki poszukiwańConsoleZamknij

Przerwij kompilowanie

☐ Shorten compiler pat

12:49

Microsoft Visual Studio 2020

The screenshot displays the Microsoft Visual Studio 2020 IDE. The main window shows the source code for 'wyklad1_1.cpp'. The code includes a simple 'main' function that prints 'Hello World!\n' and a list of tips for starting a new project. The 'Eksplorator rozwiązań' (Solution Explorer) on the right shows the project structure, including 'Odwolania' (References), 'Zależności zewnętrzne' (External Dependencies), 'Pliki nagłówkowe' (Header Files), 'Pliki zasobów' (Resource Files), and 'Pliki źródłowe' (Source Files). The 'Dane wyjściowe' (Output) window at the bottom shows the execution results, indicating that the program ran successfully and printed 'Hello World!\n'.

```
1 // wyklad1_1.cpp : Ten plik zawiera funkcję „main”. W nim rozpoczyna się i kończy wykonywanie programu.
2 //
3
4 #include <iostream>
5
6 int main()
7 {
8     std::cout << "Hello World!\n";
9 }
10
11 // Uruchomienie programu: Ctrl + F5 lub menu Debugowanie > Uruchom bez debugowania
12 // Debugowanie programu: F5 lub menu Debugowanie > Rozpocznij debugowanie
13
14 // Porady dotyczące rozpoczynania pracy:
15 // 1. Użyj okna Eksploratora rozwiązań, aby dodać pliki i zarządzać nimi
16 // 2. Użyj okna programu Team Explorer, aby nawiązać połączenie z kontrolą źródła
17 // 3. Użyj okna Dane wyjściowe, aby sprawdzić dane wyjściowe kompilacji i inne komunikaty
18 // 4. Użyj okna Lista błędów, aby zobaczyć błędy
19 // 5. Wybierz pozycję Projekt > Dodaj nowy element, aby utworzyć nowe pliki kodu, lub wybierz pozycję Projekt > Dodaj istniejący elem
20 // 6. Aby w przyszłości ponownie otworzyć ten projekt, przejdź do pozycji Plik > Otwórz > Projekt i wybierz plik sln
21
```

Output window (Dane wyjściowe):

```
Wątek 0x2bd8 zakończył działanie z kodem 0 (0x0).
„wyklad1_1.exe” (Win32): załadowano „C:\Windows\System32\kernel.appcore.dll”.
„wyklad1_1.exe” (Win32): załadowano „C:\Windows\System32\msvcrt.dll”.
„wyklad1_1.exe” (Win32): załadowano „C:\Windows\System32\rpcrt4.dll”.
Wątek 0xa34 zakończył działanie z kodem 0 (0x0).
Wątek 0x2044 zakończył działanie z kodem 0 (0x0).
Program „[13368] wyklad1_1.exe” zakończył działanie z kodem 0 (0x0).
```

na dysku HDD – ponad 8 GB dla
programowania w C++



Polskie znaki w konsoli

Ustawienia X

Ustawienia wyszukiwania

Użytkownik Obszar roboczy

Włącz synchronizowanie ustawień

Często używane

Edytor tekstu

- Kursor
- Znajdź
- Czcionka
- Formatowanie
- Edytor różnic
- Minimapa
- Sugestie

Pliki

- Pulpit
- Okno
- Funkcje
- Aplikacja
- Zabezpieczenia
- Rozszerzenia

Default Language

Domyślny tryb języka przypisywany do nowych plików. Jeśli to ustawienie zostanie skonfigurowane jako „\${activeEditorLanguage}”, używany będzie tryb języka aktualnie aktywnego edytora tekstu, o ile istnieje.

Enable Trash

☒ Przenosi pliki/foldery do kosza systemu operacyjnego (folder Kosz w systemie Windows) przy usuwaniu. Wyłączenie tego ustawienia powoduje trwałe usuwanie plików/folderów.

Encoding

Domyślne kodowanie zestawu znaków używane podczas odczytywania i zapisywania plików. To ustawienie można również skonfigurować dla poszczególnych języków.

Central European (CP 852)

UTF-16 LE utf16le

UTF-16 BE utf16be

Western (Windows 1252) windows1252

Western (ISO 8859-1) iso88591

Western (ISO 8859-3) iso88593

Western (ISO 8859-15) iso885915

Western (Mac Roman) macroman

DOS (CP 437) cp437

Arabic (Windows 1256) windows1256

Arabic (ISO 8859-6) iso88596

Baltic (Windows 1257) windows1257

Baltic (ISO 8859-4) iso88594

Celtic (ISO 8859-14) iso885914

Central European (Windows 1250) windows1250

Central European (ISO 8859-2) iso88592

Central European (CP 852) cp852

Central European (CP 852)

WYJŚCIOWE

TERMINAL

KONSOLA DEBUGOWA

C/C++ Compile Run

i

10

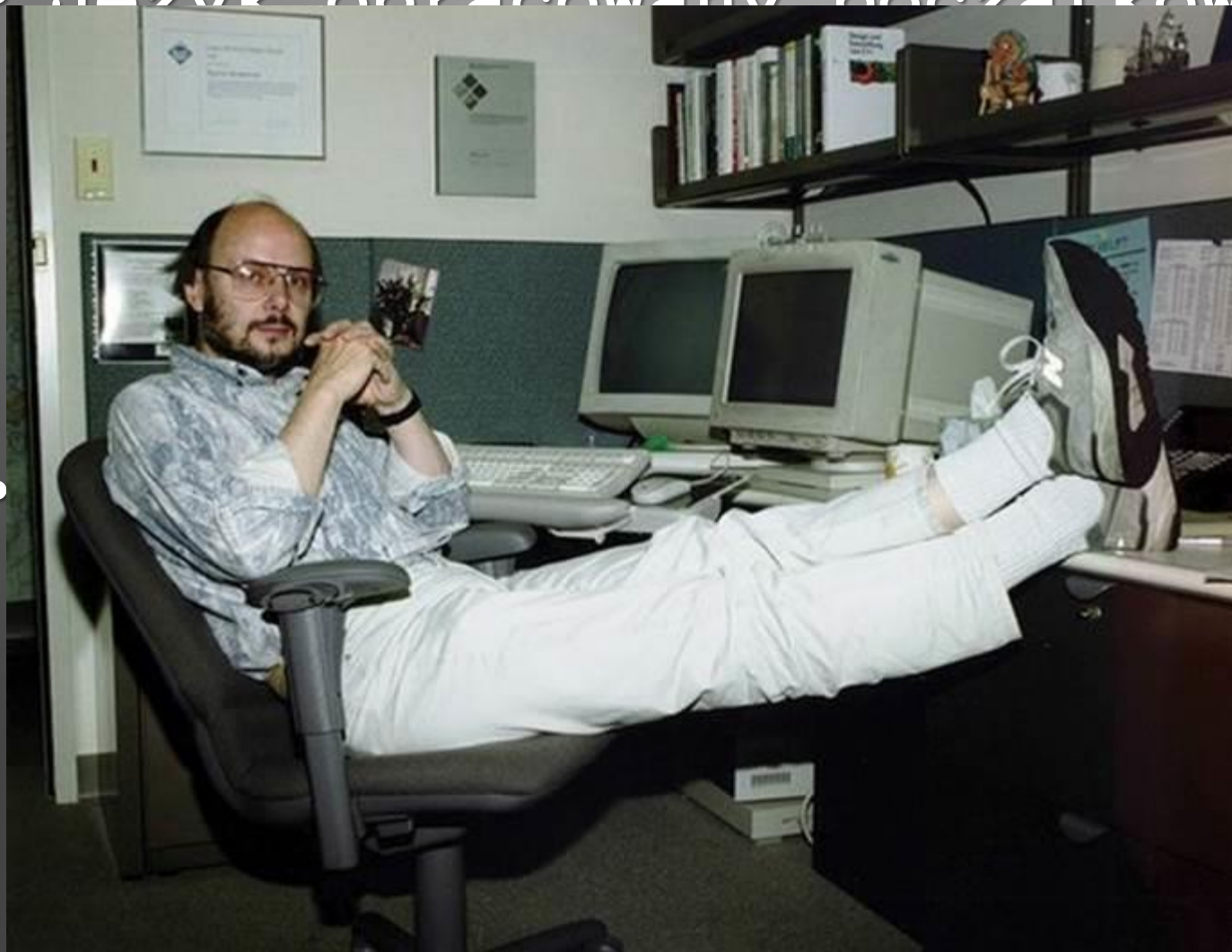
- Paradygmaty programowania.
- Programowanie strukturalne w języku C - przegląd podstawowych struktur danych i instrukcji sterujących.
- Złożone struktury danych: struktury, unie
- Obsługa wejścia-wyjścia, wykorzystanie manipulatorów.
- listy powiązane (linked lists), podwójnie powiązane listy (double linked lists)
- Zapis i odczyt danych z pliku.
- Wyrażenia regularne
- Obsługa wyjątków.



- Programowanie obiektowe w języku C++ - podstawy.
- Idea i pojęcia programowania obiektowego.
- Pojęcie klasy i obiektu.
- Przetwarzanie obiektów.
- Właściwości programowania zorientowanego obiektowo: hermetyzacja, dziedziczenie, polimorfizm.
- Funkcje i klasy zaprzyjaźnione.
- Przeciążanie operatorów i funkcji.
- Szablony funkcji i szablony klas.
- Standardowa biblioteka szablonów.
- Klasa string



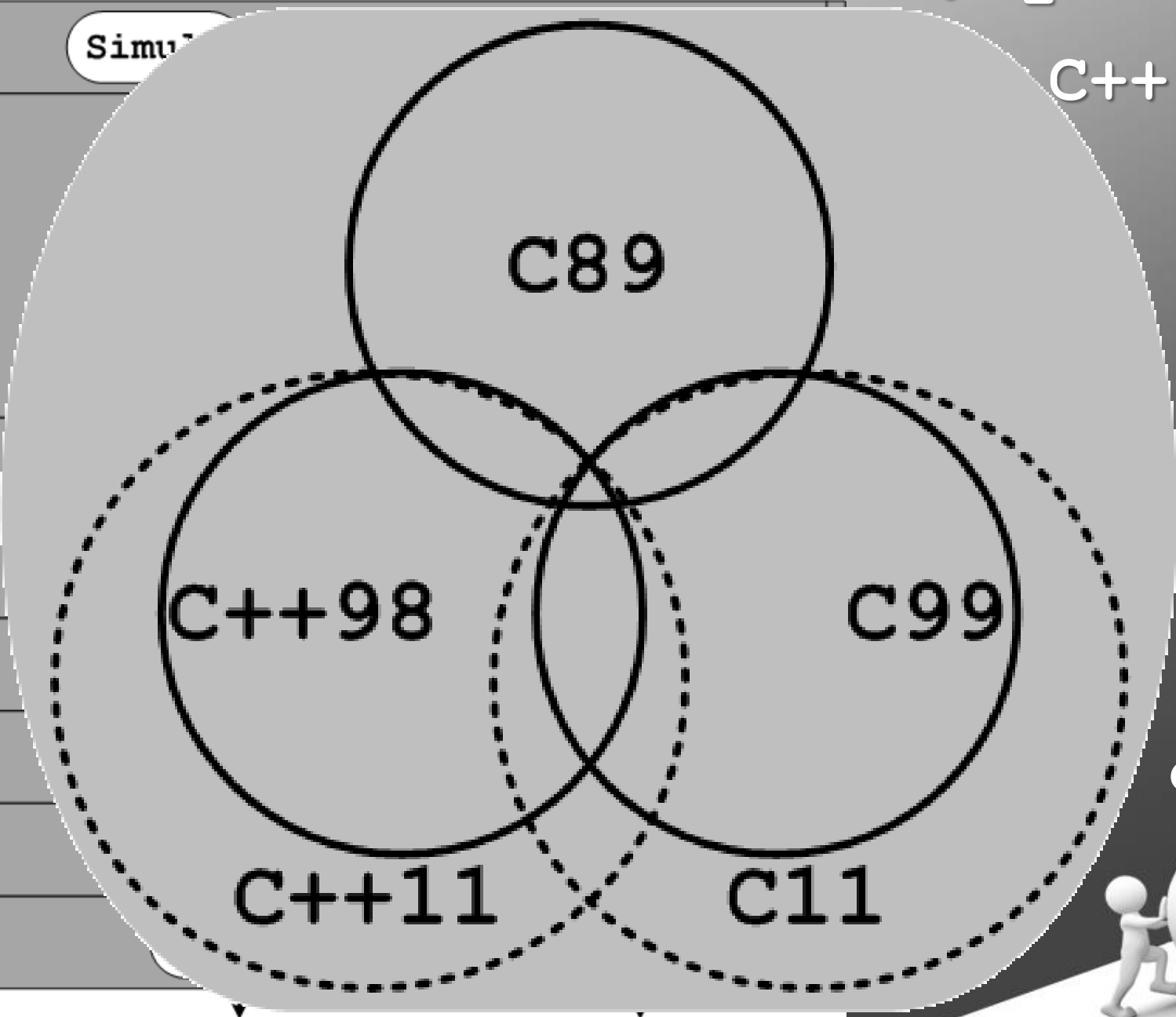
- Język opracowany początkowo w
ający
ktowego
- atoriach
zez



- Początkowo określany był jako „język C z klasami”
- Pod nazwą C++ zaczął funkcjonować od 1983 r. gdy po raz pierwszy został zastosowany poza BELL LABS
- „++” został zaczerpnięty z operacji inkrementacji **i++**, oznaczał lepszą wersję **C**
- Do 1998 r. zdefiniowany był w nieformalnym standardzie **ARM** (**A**nnotated **R**eference **M**anual)



1967	Simula
1978	
1980	
1985	
1989	
1998	
1999	
2011	



C++

ges

do



- Język C++ jest stale rozwijany, nowe wersje standardu publikowane są co około 3 lata
- Aktualna wersja **C++20**
- Znaczna część dostępnych pozycji literaturowych dotyczy wersji **C++11** (była to duża aktualizacja standardu)
- Kolejna wersja planowana **C++23**



Rok	Standard C++	Nazwa nieoficjalna	
1998	ISO/IEC 14882:1998	C++98	
2003	ISO/IEC 14882:2003	<u>C++03</u>	
2011	ISO/IEC 14882:2011	<u>C++11</u>	C++0x
2014	ISO/IEC 14882:2014	<u>C++14</u>	C++1y
2017	ISO/IEC 14882:2017	<u>C++17</u>	C++1z
2020	ISO/IEC 14882:2020	<u>C++20</u>	C++2a



Styczeń 2020	Styczeń 2019	Język programowania	Pozycja	Zmiana
1	1	Java	16.896%	-0.01%
2	2	C	15.773%	+2.44%
3	3	Python	9.704%	+1.41%
4	4	C++	5.574%	-2.58%
5	7	C#	5.349%	+2.07%
6	5	Visual Basic .NET	5.287%	-1.17%
7	6	JavaScript	2.451%	-0.85%
8	8	PHP	2.405%	-0.28%
9	15	Swift	1.795%	+0.61%
10	9	SQL	1.504%	-0.77%

TIOBE Index for January 2020



Feb 2021	Feb 2020	Programming Language	Ratings	Change
1	2	C	16.34%	-0.43%
2	1	Java	11.29%	-6.07%
3	3	Python	10.86%	+1.52%
4	4	C++	6.88%	+0.71%
5	5	C#	4.44%	-1.48%
6	6	Visual Basic	4.33%	-1.53%
7	7	JavaScript	2.27%	+0.21%
8	8	PHP	1.75%	-0.27%
9	9	SQL	1.72%	+0.20%
10	12	Assembly language	1.65%	+0.54%

TIOBE Index for February 2021



- Kod programu w języku C++ pisze się w analogicznym układzie jak w języku C
 - Nagłówek – deklaracje bibliotek, instrukcje preprocesora, deklaracje funkcji, klas, zmiennych globalnych itd.
 - Funkcja `main()` i kod pozostałych funkcji programu
- Plik z kodem źródłowym ma rozszerzenie `.cpp` a plik nagłówkowy `.hpp`



- W języku C++ podobnie jak w języku C do kodu programu podpinają się biblioteki dostarczające:
 - Typy danych
 - Funkcje
 - Klasy
- W języku C++ można korzystać z bibliotek przeznaczonych dla języka C
- Ze względu na przyjętą nomenklaturę nazwy tych biblioteki rozpoczynają się od litery '**c**' i nie posiadają rozszerzenia '**.h**'



C	C++	C	C++	C	C++
assert.h	cassert	ctype.h	cctype	errno.h	cerrno
float.h	cfloat	iso646.h	ciso646	limits.h	climits
locale.h	clocale	math.h	cmath	setjmp.h	csetjmp
signal.h	csignal	stdarg.h	cstdarg	stddef.h	cstddef
stdio.h	cstdio	stdlib.h	cstdlib	string.h	cstring
time.h	ctime	wchar.h	wchar	ctype.h	cctype



Podstawowa biblioteka

- W języku C++ podstawową biblioteką wykorzystywaną w niemal każdym programie jest biblioteka wejścia-wyjścia `<iostream>`
- Biblioteka dostarcza funkcji obsługujących strumień wejścia i wyjścia
- Większość bibliotek w C++ dostarcza klasy umożliwiające tworzenie obiektów danego typu oraz dostarczające metody do ich obsługi



Klasa `ios`:
Ogólne właściwości
strumienia, zawiera
wskaźniki do obiektu
klasy `streambuf`

Klasa `streambuf`:
zarządza pamięcią
buforów wejściowych
i wyjściowych

Klasa `istream`
metody do obsługi
strumienia wejściowego

Klasa `ostream`
metody do obsługi
strumienia wyjściowego

Klasa `iostream` dziedziczy
metody do obsługi
wejścia i wyjścia z klas
`istream` i `ostream`



- Komunikacja z użytkownikiem w programach konsolowych realizowana jest za pomocą metod odwołania do właściwego strumienia oraz operatorów przekierowania danych
- Strumień wyjściowy realizowany jest za pośrednictwem metody **cout**
- Strumień wyjściowy obsługiwany jest za pomocą metody **cin**
- Operatory przekierowania **<<** i **>>** określają kierunek przesyłania danych



```
1  #include <iostream>
2
3  int main()
4  {
5      std::cout << "Hello world!" << std::endl;
6  }
```

- **std** – oznacza odwołanie do standardowych elementów języka C++
- **::** – jest operatorem zasięgu, za pomocą którego odwołujemy się do metod z przestrzeni podanej przez dwukropkami
- **endl** – realizuje analogiczne zadanie do **"\n"** w języku C

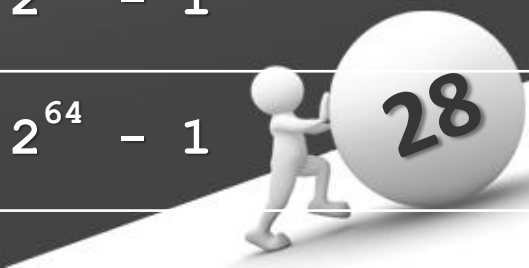


- W języku C++ zmienne deklaruje się w sposób analogiczny jak w C
- Podaje się typ danych a następnie nazwę zmiennej
- Można inicjować zmienną przypisując po deklaracji wartość danego typu do zmiennej
- Podstawowe typy danych można modyfikować podobnie jak w języku C



Typ całkowity

Nazwa	Wielkość (bajty)	Zakres
short	2	$-2^{15} \div 2^{15} - 1$
int	4	$-2^{31} \div 2^{31} - 1$
long	4	$-2^{31} \div 2^{31} - 1$
long long	8	$-2^{63} \div 2^{63} - 1$
unsigned short	2	$0 \div 2^{16} - 1$
unsigned int	4	$0 \div 2^{32} - 1$
unsigned long	4	$0 \div 2^{32} - 1$
unsigned long long	8	$0 \div 2^{64} - 1$



Typ rzeczywisty

Nazwa	Wielkość (bajty)	Zakres
float	4	pojedyncza precyzja - dokładność 6 - 7 cyfr po przecinku - $3.4 \cdot 10^{\pm 38}$
double	8	podwójna precyzja - dokładność 15 - 16 cyfr po przecinku - $1.7 \cdot 10^{\pm 308}$
long double	12	liczby z ogromną dokładnością - 19 - 20 cyfr po przecinku - $1.7 \cdot 10^{\pm 308}$



Typ znakowy i logiczny

Nazwa	Wielkość (bajty)	Zakres
char	1	-128 ÷ 127
unsigned char	1	0 ÷ 255

Nazwa	Wielkość (bajty)	Zakres
bool	1	true (1) false (0)



```
1 #include <iostream>
```

C:\WINDOWS\system32\cmd.exe

Typy zmiennych:

```
int x=25
```

```
float y=3.1415
```

```
char z=S
```

```
bool w=1
```

```
1
```

```
1
```

```
1 Press any key to continue . . .
```

```
1
```

```
14 }
```



- $x+y$ //dodawanie
- $x-y$ //odejmowanie
- $x*y$ //mnożenie
- x/y //dzielenie
- $x\%y$ //reszta z dzielenia (int)



```
1  #include <iostream>
2      x=0 y=1
3  int main(x=1 y=2
4  {      x=2 y=3
5      int x x=3 y=4
6      for(i x=4 y=5
7      {      s x=5 y=6
8  }      x=6 y=7
9      x=7 y=8
10     x=8 y=9
11     x=9 y=10
12     Press any key to continue . . .
```

<< std::endl;



Inicjator auto

- Inicjując zmienne można skorzystać z inicjacji automatycznej
- Kompilator dobiera typ dla zmiennej na podstawie przypisanej do niej wartości

```
auto x = 10;           //typ int
auto y = false;        //typ bool
auto z = 'a';          //typ char
auto v = sqrt(x);      //typ double
auto w = 1.34;         //typ float
```



- W języku C++ pojęcie stałej ma szersze znaczenie jak W C
- W C++ wyróżnia się wartość stałej zwracanej przez wyrażenie, funkcję lub metodę
- **const** – przedrostek deklaracji stałej „zmiennej”
- **constexpr** – przedrostek wyrażenia, funkcji lub metody zwracającej wartość stałą



```
1  #include <iostream>
2  #define w 15;
3  constexpr int iloczyn(const int a, int b)
4  {
5      return a*b;
6  }
7  int main()
8  {
9      const int x = 5;
10     int y = 10;
11     constexpr int z1 = x*x;
12     constexpr int z2 = x*y; // x*w
13     const int z3 = x*y;
14     const int z4 = iloczyn(x,y);
15     std::cout << z1 << "," << z2 << "," << z3 << "," << z4;
16 }
```



Złożone typy danych

```
1  #include <iostream>
2  using namespace std;
3  struct test{
4      int A,B;
5  };
6  union test2
7  {
8      int A;
9      float B;
10 };
11
12 int main()
13 {
14     test alfa;
15     test2 beta;
16     cout << "Podaj dane:" << endl;
17     cout << "A=";
18     cin >> beta.A;
19     cout << "B=";
20     cin >> beta.B;
21     cout << "Podano A=" << beta.A << " i B=" << beta.B;
22 }
```

ować

e z C) –

ymaga

słowa

on

znie przy



```
C:\WINDOWS\system32\cmd.exe

Typy zmiennych:
Podaj wartosc typu int x=11
Podaj wartosc typu float y=123.234
Podaj wartosc typu bool z=13
Wprowadzono x=11 y=123.234 z=1

Press any key to continue . . .
```



- W języku C++ wprowadzono mechanizm ułatwiający pisanie dużych programów przez wielu programistów
- Możliwe jest grupowanie zmiennych, funkcji, klas, ... w tak zwane przestrzenie nazw
- W ramach różnych przestrzeni nazw w jednym programie mogą funkcjonować elementy o tych samych nazwach
- Przestrzenie nazw deklaruje się w obszarze nagłówkowym



Przestrzenie nazw

12:49

```
1  #include <iostream>
2  namespace calk
3  {
4      int x=10;
5      void write(){std::cout << "Zmienna x=" << x << std::endl;}
6  }
7  namespace rzecz
8  {
9      float x=3.1415;
10     void write(){std::cout << "Zmienna x=" << x << std::endl;}
11 }
12 namespace inne
13 {
14     char x='x';
15     bool y=true;
16 }
17 int main()
18 {
19     using rzecz::write;
20     using namespace inne;
21     calk::write();
22     write();
23     std::cout << "x=" << x << " y=" << y << std::endl;
24 }
```

C:\WINDOWS\system32\cmd.exe

Zmienna x=10

Zmienna x=3.1415

x=x y=1

Press any key to continue . . .



Przestrzeń standardowa

- Większość podstawowych typów i funkcji w języku C++ jest zgrupowana w standardowej przestrzeni nazw
- W małych programach można ustawić domyślną przestrzeń

using namespace std;

- Dzięki temu nie trzeba przed funkcjami i typami z przestrzeni **std** umieszczać odwołania **std::**
- Nie jest to jednak zalecane



```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int x;
6      float y;
7      bool z;
8      cout << "Typy zmiennych:" << endl;
9      cout << "Podaj wartosc typu int x=";
10     cin >> x;
11     cout << "Podaj wartosc typu float y=";
12     cin >> y;
13     cout << "Podaj wartosc typu bool z=";
14     cin >> z;
15     cout << "Wprowadzono x=" << x << " y=" << y << " z=" << z << endl;
16 }
```



```
1  #include <iostream>
2
3
4
5
6
7
8
9
10
11
12
13
14
```

C:\WINDOWS\system32\cmd.exe

Hello World!

tab[5]=5

Twoje imie to: ☺

Podaj imie: Michał

Twoje imie to: Michał

Press any key to continue . . .

```
    dl;
    dl;
```

- Tablice można deklarować i inicjować



- Podejmowanie decyzji w języku C++ realizowane może być za pomocą instrukcji warunkowej `if()else` i wyboru `switch()case`
- Zasady ich użycia są identyczne z tymi poznanymi w języku C



Operatory porównania i logiczne

<code>x==y</code>	//równy
<code>x!=y</code>	//różny
<code>x<y</code>	//mniejszy
<code>x>y</code>	//większy
<code>x<=y</code>	//mniejszy lub równy
<code>x>=y</code>	//większy lub równy
<code>&&</code>	//iloczyn i (AND)
<code> </code>	//suma lub (OR)
<code>!</code>	//negacja (NOT)



12:49

```
1  #include <iostream>
2  #include <cmath>
3  int main()
4  {
5      C:\WINDOWS\system32\cmd.exe
6      Podaj liczbę całkowita: -45
7      Moduł x=45
8      Wybierz działanie:
9      a) kwadrat liczby
10     b) pierwiastek kwadratowy liczby
11     :.> a
12     x^2=2025
13
14     Press any key to continue . . .
15
16     default:
17     std::cout << "Błędny wybór opcji programu." << std::endl;
18
19 }
20
21
22
23
24
25 }
```



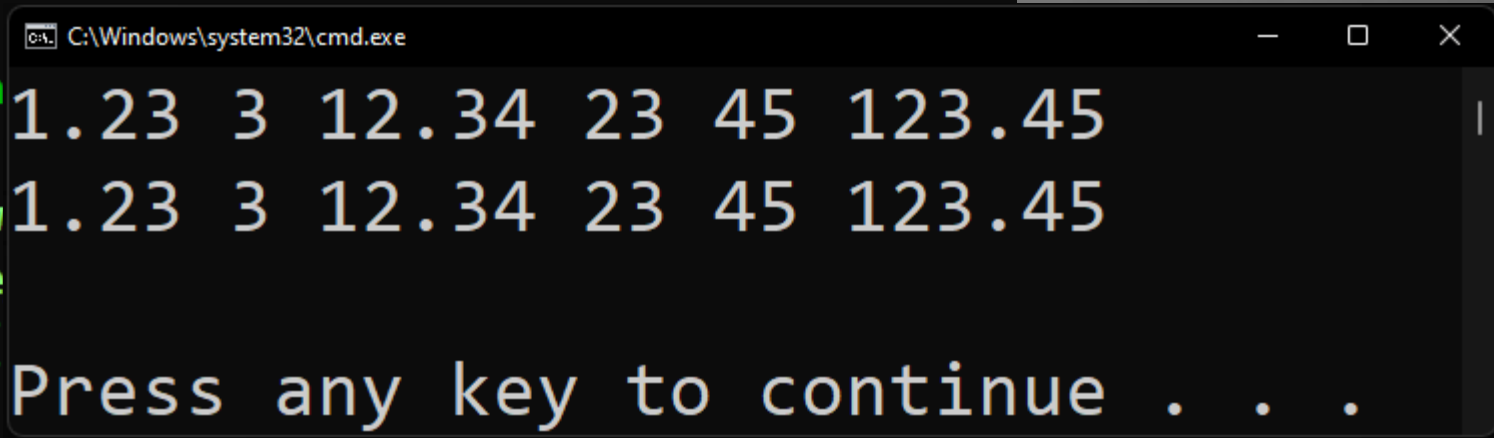
- Pętla iteracyjna **for()**
- Pętla warunkowa **while()**
- Pętla warunkowa **do..while()**
- Instrukcje przerywania pętli:
 - **continue;**
 - **break;**
 - **return;**



Pętla **for()** po tablicy

Pętla iteracyjna `for()` w języku C++ może zostać zdefiniowana do iteracji po elementach pętli bez konieczności indeksowania

```
1  #include <iostream>
2
3  int main
4  {
5      float tab[6] = {1.23, 3, 12.34, 23, 45, 123.45};
6      for(
7
8
9      std::cout << std::endl;
10     for(auto i:tab)
11         std::cout << i << " ";
12     std::cout << std::endl;
13 }
```



```
C:\Windows\system32\cmd.exe
1.23 3 12.34 23 45 123.45
1.23 3 12.34 23 45 123.45
Press any key to continue . . .
```



- Każdy program w języku C/C++ składa się z szeregu powiązanych ze sobą funkcji
- Główną funkcją każdego programu jest **main()**
- Pisząc kod programu poszczególne jego zadania zazwyczaj wydziela się do funkcji które są następne wywoływane zgodnie z przewidzianym algorytmem działania



- Każda funkcja (poza `main()`) powinna zostać najpierw zadeklarowana
- Deklaracja zawiera informacje o:
 - typie danych zwracanym przez funkcję,
 - nazwie funkcji,
 - liście typów parametrów wejściowych.
- Język C++ pozwala na przeciążanie funkcji



12:49

```
1  #include <iostream>
2  int dodaj(int,int);
3  float dodaj(float,float);
4  flo
5  flo
6  in
7  {
8      x+y=-210.994
9      x+b=33.456
10     a+y=-229.45
11
12     Press any key to continue . . .
13
14 }
15 float dodaj(float x, float y){ return x+y;}
16 int dodaj(int x, int y){ return x+y;}
17 float dodaj(int x, float y){ return x+y;}
18 float dodaj(float x, int y){ return x+y;}
```



Funkcja `main()`

- Parametry wejściowe tak jak każda funkcja deklarowana w C++
- Funkcja `main()` może posiadać dwa parametry
- Pierwszy typu `int` przechowuje informacje o liczbie parametrów
- Drugi jest wskaźnikiem na tablicę znakową i przechowuje ciągi tekstowe poszczególnych parametrów
- Pierwszym parametrem jest nazwa programu – indeks `0`



Funkcja main()

```
1  #include <iostream>
2
3  int main(int numer, char *param[])
4  {
5      for(int i=0;i<numer;i++)
6          std::cout << "parametr " << i << ": " << param[i] << std::endl;
7
8  }
```

C:\Windows\system32\cmd.exe

```
parametr 0: .\kod7.exe
parametr 1: Alfa
parametr 2: 23
parametr 3: Ala ma kota
parametr 4: 23.123
```

Press any key to continue . . .



Plik nagłówkowy

- Podobnie jak w języku C kod programu można podzielić na kilka osobnych plików
- Podstawowym podział jest realizowany na plik nagłówkowy (**.hpp**) i źródłowy (**.cpp**)
- Kod programu można podzielić zadaniowo i zapisać w osobnych parach (**.hpp** i **.cpp**)
- Plik nagłówkowy zawiera instrukcje preprocesora (podpięcie bibliotek, plików klas), deklaracji zmiennych i stałych globalnych oraz deklaracje funkcji



```
• kod6.cpp • kod6.hpp
wyklad_1 > kod6.cpp > ...
1  #include "kod6.hpp"
2  int main()
3  {
4      dane();
5      std::cout << "Suma: " << suma();
6  }
7  void dane()
8  {
9      srand(time(0));
10     for(int i=0;i<20;i++)
11         x[i] = rand()%21;
12 }
13 int suma()
14 {
15     int n=0;
16     for(int i=0;i<20;i++)
17         n+=x[i];
18     return n;
19 }
```

```
• kod6.hpp ×
wyklad_1 > kod6.hpp > ...
1  #ifndef KOD6_HPP
2  #define KOD6_HPP
3
4  #include <iostream>
5  #include <ctime>
6  #include <cstdlib>
7  int suma();
8  void dane();
9  int x[20];
10
11 #endif
```

Złożone typy danych

- Znane z języka C złożone typy danych:
 - **struct** – struktura
 - **union** – unia
 - **enum** – typ wyliczeniowy
- **klasa** – deklaracyjny typ obiektowy



Metody wejścia i wyjścia

```
1  #include <iostream>
2
3  int main()
4  {
5      int x[] = {2,3,4,5,12,43,121};
6      char tekst[]="Ala ma kota";
7      std::cout.put(230);
8      std::cout.put( '\t' ).put( 'A' ).put( '\n' );
9      for(auto i:x)
10         std::cout.write((char*)&i,sizeof(i));
```

C:\Windows\system32\cmd.exe

Š A

☹ ♥ ♦ ♣ ♀ + y

Ala ma kota

Press any key to continue . . .

| (tekst));



Wyjściowy system liczbowy

```
1  #include <iostream>
2
3  int main()
4  {
5      float x[] = {24.34,12.3,4.34,54.5,1.22,43.43,.121};
6      for(auto i:x){
7          std::dec(std::cout);
8          std::cout << i << "\t";
```

te do
można
pomocą
liczba

C:\Windows\system32\cmd.exe

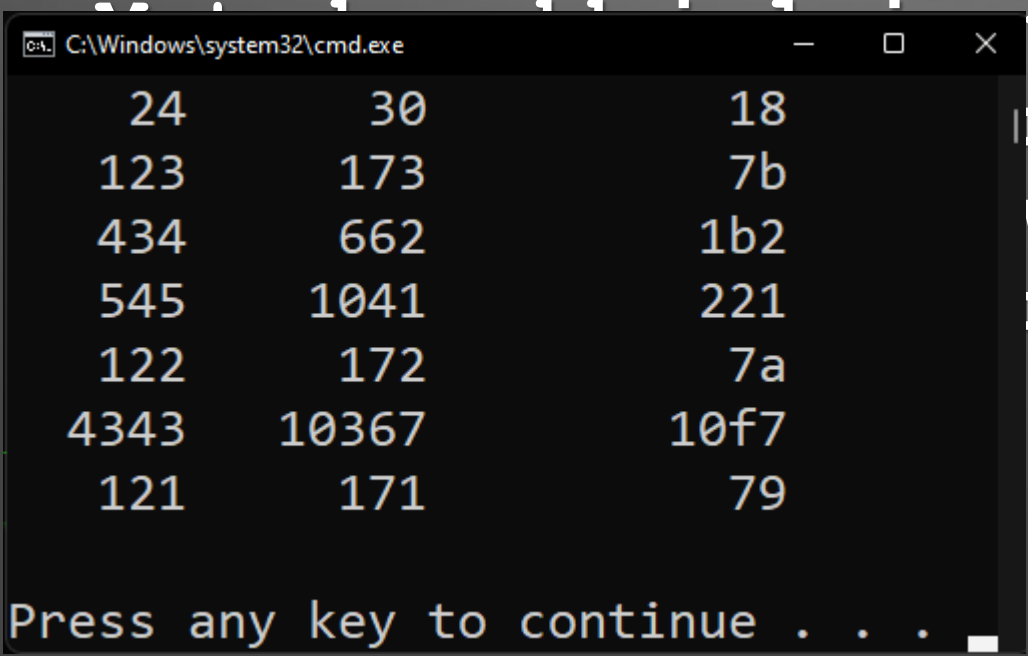
```
24.34    24.34    24.34
12.3     12.3     12.3
4.34     4.34     4.34
54.5     54.5     54.5
1.22     1.22     1.22
43.43    43.43    43.43
0.121    0.121    0.121
```

Press any key to continue . . .



Szerokość pola danych

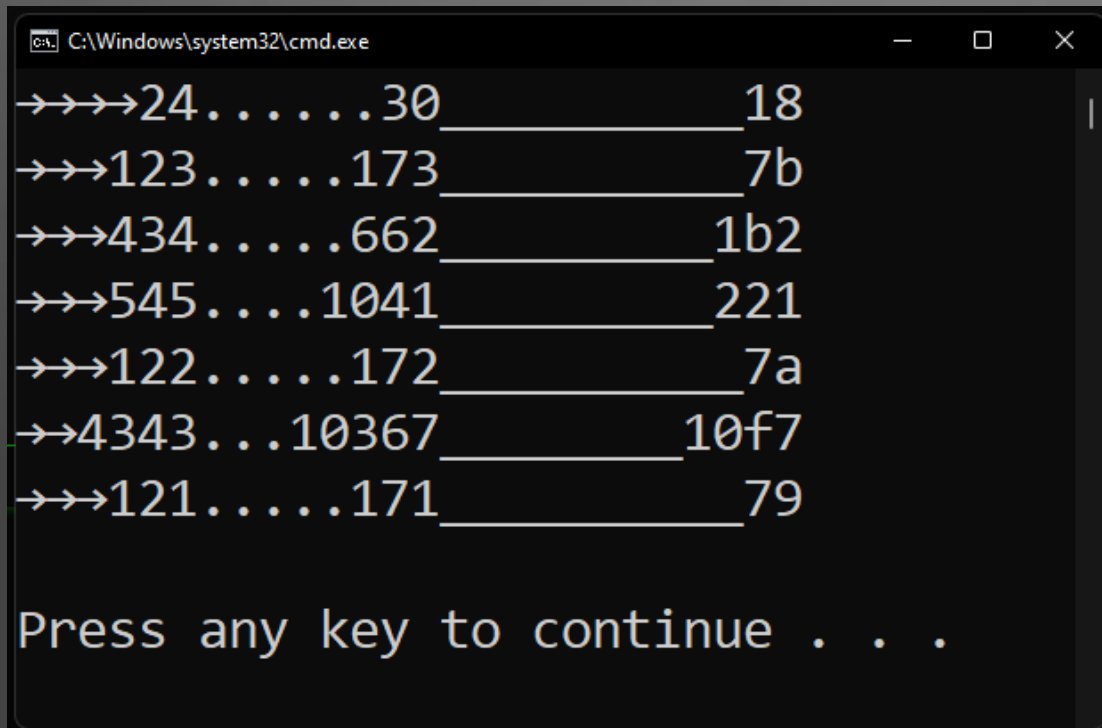
```
1  #include <iostream>
2
3  int main()
4  {
5      int x[] = {24,123,434,545,122,4343,121};
6      for(auto i:x){
7          std::dec(std::cout); std::cout.width(6); std::cout << i;
8          std::cout << std::oct; std::cout.width(8); std::cout << i;
9          std::cout << std::hex; std::cout.width(12); std::cout << i <<std::endl;
10     }
11 }
```



tylko na
nych wysyłany do
wego, następny
ny w sposób



```
1  #include <iostream>
2
3  int main()
4  {
5      int x[] = {24,123,434,545,122,4343,121};
6      for(auto i:x){
7          std::dec(std::cout); std::cout.fill(26); std::cout.width(6); std::cout << i;
8          std::cout << std::oct; std::cout.width(8); std::cout.fill('.'); std::cout << i;
9          std::cout.fill('_'); std::cout << std::hex; std::cout.width(12); std::cout << i <<std::endl;
10     }
11 }
```



składowej
nienić znak
pól



float - precyzja

```
C:\Windows\system32\cmd.exe

precyzja(6)          3.14159          3.14159
precyzja(7)          3.141593         3.141593
precyzja(8)          3.1415927        3.1415926
precyzja(9)          3.14159265       3.14159264
precyzja(10)         3.141592654      3.141592644
precyzja(11)         3.1415926536     3.1415926436
precyzja(12)         3.14159265359    3.14159264359
precyzja(13)         3.14159265359    3.14159264359
precyzja(14)         3.1415926535898  3.1415926435898
precyzja(15)         3.14159265358979 3.14159264358979
precyzja(16)         3.141592653589793 3.141592643589794
precyzja(17)         3.1415926535897931 3.1415926435897941
precyzja(18)         3.14159265358979312 3.14159264358979408
precyzja(19)         3.141592653589793116 3.141592643589794078
precyzja(20)         3.141592653589793116 3.1415926435897940784

Press any key to continue . . .

17      std::cout << pi2 << std::endl;
18      }
```

$\pi \approx 3,141592653589793238462643383279502884197169...$



Sterowanie wyświetlaniem

- Precyzyjne sterowanie sposobu wyświetlania różnych wartości służy funkcja składowa `.setf()`
- Usuwanie nadanego formatowania realizuje metoda `.unsetf()`
- Funkcja ta korzysta ze stałych zdefiniowanych w klasie `ios_base`
- Konieczne jest odwołanie standardowego wejścia do klasy `ios_base`

```
using std::ios_base
```

```
lub
```

```
using namespace std;
```



Sterowanie wyświetlaniem

- `ios_base::boolalpha` – wartości logiczne wyświetlane są jako **true** i **false**
- `ios_base::showbase` – wyświetla przedrostek systemu liczbowego
- `ios_base::showpoint` – wyświetla „.” oddzielającą część ułamkową (wraz z zerami)
- `ios_base::uppercase` – przy zapisie hex korzysta z dużych liter
- `ios_base::showpos` – przy liczbach dodatnich wyświetla „+”



```
1  #include <iostream>
2
3  C:\WINDOWS\system32\cmd.exe
4  Test zmiennej bool: 1
5
6  Test zmiennej bool: true
7
8  Test zmiennej int(hex): 8a7
9
10 Test zmiennej int(HEX): 8A7
11
12 Test zmiennej int(HEX): 0X8A7
13
14 Test zmiennej int(hex): 0x8a7
15
16 Press any key to continue . . .
17
18     std::cout.unsetf(ios_base::uppercase);
19     std::cout << "Test zmiennej int(hex): " << i << std::endl;
20 }
```



Funkcja **setf()** dla pewnych stałych może przyjmować drugi parametr (stała) – uszczegółwiający

```
cout.setf(st.szczegolowa, st.glowna);
```

ios_base::floatfield	ios_base::fixed	Notacja stałoprzecinkowa
	ios_base::scientific	Notacja naukowa
ios_base::adjustfield	ios_base::left	Wyrównanie do lewej
	ios_base::right	Wyrównanie do prawej
	ios_base::internal	Wyrównanie do lewej dla znaku lub przedrostka systemu liczbowego, wyrównanie do prawej dla wartości



```
1  #include <iostream>
2
3  int main()
4  {
5      using std::ios_base;
6      int X = 23687;
7      ios_base::fmtflags styl = ios_base::uppercase;
8      std::cout.setf(styl);
9      std::cout << "X (HEX) = ";
10     std::cout << std::hex;
11     std::cout << X << std::endl;
```

C:\Windows\system32\cmd.exe

X (HEX) = 5C87

X (hex) = 5c87

Press any key to continue . . .

z

na
i

ie
ć



- W zależności od stosowanego kompilatora dostępne są *manipulatory* realizujące wybrane zadania ustawiane za pomocą funkcji `setf()`

```
cout << manipulator;
```

- Jeżeli dany manipulator jest niedostępny konieczne jest skorzystanie z funkcji `setf()`



<code>boolalpha</code>	<code>setf(ios_base::boolalpha)</code>
<code>noboolalpha</code>	<code>unsetf(ios_base::boolalpha)</code>
<code>showbase</code>	<code>setf(ios_base::showbase)</code>
<code>noshowbase</code>	<code>unsetf(ios_base::showbase)</code>
<code>showpoint</code>	<code>setf(ios_base::showpoint)</code>
<code>noshowpoint</code>	<code>unsetf(ios_base::showpoint)</code>
<code>showpos</code>	<code>setf(ios_base::showpos)</code>
<code>noshowpos</code>	<code>unsetf(ios_base::showpos)</code>
<code>uppercase</code>	<code>setf(ios_base::uppercase)</code>
<code>nouppercase</code>	<code>unsetf(ios_base::uppercase)</code>



<code>internal</code>	<code>setf(ios_base::internal, ios_base::adjustfield)</code>
<code>left</code>	<code>setf(ios_base::left, ios_base::adjustfield)</code>
<code>right</code>	<code>setf(ios_base::right, ios_base::adjustfield)</code>
<code>dec</code>	<code>setf(ios_base::dec, ios_base::basefield)</code>
<code>hex</code>	<code>setf(ios_base::hex, ios_base::basefield)</code>
<code>oct</code>	<code>setf(ios_base::oct, ios_base::basefield)</code>
<code>fixed</code>	<code>setf(ios_base::fixed, ios_base::floatfield)</code>
<code>scientific</code>	<code>setf(ios_base::scientific, ios_base::floatfield)</code>



W bibliotece `<iomanip>` dostępne są manipulatory realizujące tożsame zadanie z funkcjami:

- `cout.fill(char);`
`cout << setfill(char) << ...`
- `cout << fixed;`
`cout.precision(int);`
`cout << setprecision(int) << ...`
- `cout.width(int);`
`cout << setw(int) << ...`



```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  int main()
5  {
6      const char * osoby[] = {"Jan Kowalski", "Adam Wiśniewski", "Ola Kwiatek"};
7      const int premia[] = {1000, 500, 1245};
8      for(int i=0; i<3; i++)
9      {
10         cout.fill('_');
11         cout.width(18);
12         cout << osoby[i] << " ";
13         cout.fill('.');
14         cout.width(8);
15         cout << premia[i] << " zł\n";
16     }
17     cout << endl;
18     for(int i=0; i<3; i++)
19     {
20         cout << setfill('_') << " ";
21         cout << setfill('.') << " ";
22     }
23 }
```

```
C:\Windows\system32\cmd.exe

_____Jan Kowalski:.....1000 zł
_____Adam Wiśniewski:.....500 zł
_____Ola Kwiatek:.....1245 zł

_____Jan Kowalski:.....1000 zł
_____Adam Wiśniewski:.....500 zł
_____Ola Kwiatek:.....1245 zł

Press any key to continue . . .
```



Strumień wyjściowy

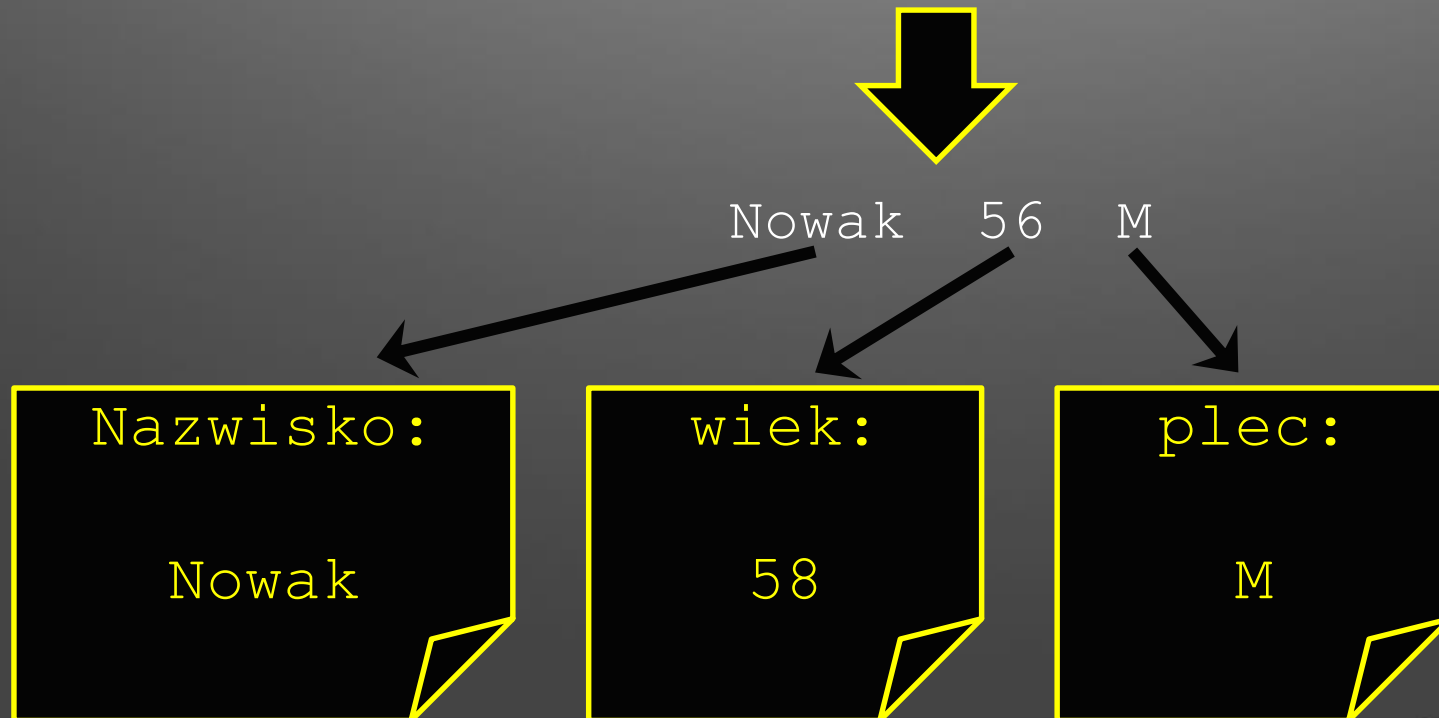
- Obiekt `cin` reprezentuje wejściowy strumień bitowy
- Standardowy strumień wejściowy generowany jest przez klawiaturę
- Dane ze strumienia wejściowego operatorem przypisania są konwertowane na typ żądany przez zmienną do której jest skierowany



Strumień wejściowy

```
char nazwisko[20];  
int wiek;  
char plec;  
cin >> nazwisko >> wiek >> plec;
```

Pomijane są odstępy, znaki nowego wiersza i tabulacji



Strumień wejściowy

```
1  #include <iostream>
```

```
2  int suma()
```

```
3  {
```

```
4      int sum=0,x;
```

```
5      while(std::cin >> x)
```

```
6      {
```

```
7          sum+=x;
```

```
8      }
```

```
9      return sum;
```

```
10 }
```

```
11 int main()
```

```
12 {
```

```
13     std::cout << "Dane wejściowe: ";
```

```
14     int sum = suma();
```

```
15     std::cout << "Suma=" << sum << std::endl;
```

```
16     std::cout << "Nowe dane wejściowe: ";
```

```
17     sum = suma();
```

```
18     std::cout << "Suma=" << sum << std::endl;
```

```
19 }
```

C:\Windows\system32\cmd.exe

Dane wejściowe: 1 2 3 4 5 a 23

Suma=15

Nowe dane wejściowe: Suma=0

Press any key to continue . . .



- Generator liczb pseudolosowych z języka C
 - Wymaga bibliotek `<stdlib.h>` i `<ctime>`
 - Ustawienie ziarna generatora –
`srand(time(0))` `srand(time(NULL))`
 - Generowanie w zakresie 0-RAND_MAX
`rand()` – `RAND_MAX = 32767`
 - Generowanie w zakresie (a,b)
`a+rand() % (b-a+1)`



- Generator niedeterministyczny w języku C++

- Wymaga biblioteki `<random>`

- deklaracja generatora:

```
random_device nazwa;
```

- Wybór silnika generatora:

```
typ nazwaSilnika(nazwa());
```

- `default_random_engine` – silnik domyślny
 - `minstd_rand` – minimalny standardowy generator
 - `minstd_rand0` – minimalny standardowy generator
 - `mt19937` – generator Mersenne Twister 19937
 - `mt19937_64` – generator Mersenne Twister 19937
 - `ranlux24_base` – generator Ranlux o podstawie 24
 - `ranlux48_base` – generator Ranlux o podstawie 48
 - `ranlux24` – generator Ranlux 24
 - `ranlux48` – generator Ranlux 48
 - `knuth_b` – generator Knuth-B



- Generator niedeterministyczny w języku C++

- Możliwy wybór rozkładu

- Rozkład całkowity `<int>`

```
uniform_int_distribution<int> nazwaR(a,b);
```

- Rozkład rzeczywisty `<float>`~~`<double>`~~

```
uniform_real_distribution<float> nazwaR(a,b);
```

- Generowanie liczby losowej

```
x = nazwaSilnika();
```

```
x = nazwaR(nazwaSilnika);
```



Przykład

13:24

```
C:\WINDOWS\system32\cmd.exe

Liczba losowa: 5842443
Liczba z rozkładu całkowitego: -2.58412
Liczba z rozkładu rzeczywistego: 8
-----
Liczba losowa: 13309813
Liczba z rozkładu całkowitego: -5.06122
Liczba z rozkładu rzeczywistego: 10
-----
Liczba losowa: 7223712
Liczba z rozkładu całkowitego: 2.56144
Liczba z rozkładu rzeczywistego: 4
-----
Liczba losowa: 8218192
Liczba z rozkładu całkowitego: -9.06032
Liczba z rozkładu rzeczywistego: 10
-----
Liczba losowa: 4765480
Liczba z rozkładu całkowitego: 3.32433
Liczba z rozkładu rzeczywistego: -10
-----

Press any key to continue . . .
```

```
adInt(generator) << endl;
kladFloat(generator) << endl;
<< endl;
```

